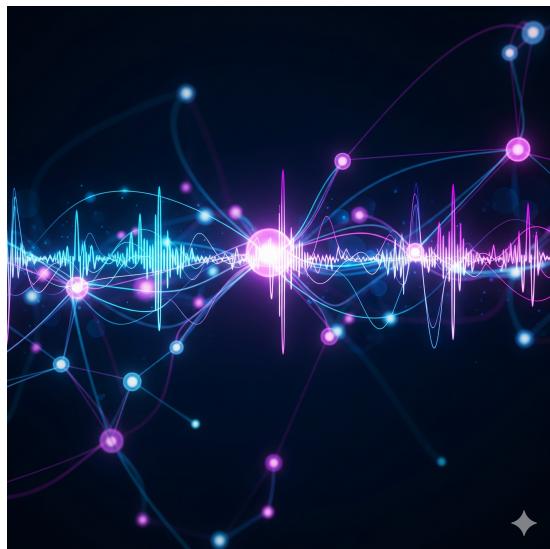


Temas Selectos de Deep Learning: Análisis Multimodal para MIR | Tarea #1

Redes Neuronales, Regularización y Aplicaciones en
MIR

César M. Aguirre Calzadilla



Centro de Investigación en Matemáticas

Maestría en Cómputo Estadístico

Catedráticos:

Dr. Victor Muñiz Sánchez

15 de septiembre de 2025

Tabla de contenidos

	Página
Ejercicio #1 Broadcasting	2
Teoría	2
Ejemplo de Broadcasting	3
Código en Numpy	4
Ejercicio #2 Redes Neuronales Multicapa	5
Ventajas de la Red A	5
Ventajas de la Red B	6
Ejercicio #3 Regularización L^1	7
Aproximación Cuadrática para Regularización L^1	7
Sparcity provocado por L^1	8
Ejercicio #4 Backpropagation	11
Teoría	11
Resumen de código	20
Resultados	20
Conclusiones	20
Ejercicio #1 Descripción del corpus	21
Teoría	21
Resumen de código	21
Resultados	21
Conclusiones	21
“Una nota personal sobre m’usica, sonido y electrónica”	22

Ejercicio #1 | Broadcasting

Calcula lo siguiente:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} + (7 \ 9)$$

Usa broadcasting de tal forma que la operación esté bien definida. Antes, averigua y describe qué es broadcasting, en el contexto de numpy.

Teoría

Bajo el contexto de Numpy, Broadcasting es un mecanismo por el cual se permite operar sobre arreglos de tal manera que si el arreglo no cumple con las dimensiones adecuadas para, por ejemplo, una suma de matrices, entonces se replica virtualmente a la forma del arreglo de mayor tamaño. Esto lo realiza sin copiar datos, de tal modo que realiza operaciones elemento a elemento de forma vectorizada, como es la intención de Numpy.

Esto lo podemos pensar de una manera matemática. Sea x un arreglo de forma (a_1, a_2, \dots, a_m) y sea y otro arreglo de la forma (b_1, b_2, \dots, b_n) , podemos alinear "por el final" (i.e. comparamos sus formas de derecha a izquierda). Si una forma tiene menos dimensiones que la otra, se considera que está rellena a la izquierda con dimensiones de tamaño 1 hasta igualar la longitud de la otra). En cada eje alineado, las dimensiones son compatibles si tenemos los siguientes casos:

$$a_i = b_i \quad \text{o} \quad a_i = 1 \quad \text{o} \quad b_i = 1$$

Si todas las dimensiones resultan compatibles, entonces la forma del resultado es la máxima por eje:

$$(\max(a'_1, b'_1), \dots, \max(a'_k, b'_k))$$

Donde a' y b' son los tuplos alineados. En caso de que algún eje no cumpla con la regla, entonces hay un error de broadcasting.

Este método es realmente útil ya que no crea copias del arreglo de menor dimensión para igualar las de los arreglos más grandes, sino que "simula" que el arreglo de menor dimensión hace match con las dimensiones del más grande. Esto le permite a Numpy ahorrar mucho tiempo de cómputo, sobre todo cuando pensamos en operaciones de conjuntos de datos gigantes, como lo serían datasets de cientos de miles de imágenes, de documentos, o

de audio. Podríamos incluso pensar en algo más básico, un vector de 10,000 elementos para sumarlo a una matriz de 10000×10000 , con broadcasting no utilizaríamos tanta memoria al no duplicar manualmente ese vector las 10,000 veces.

Ejemplo de Broadcasting

La tarea nos propone el ejemplo siguiente:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} + (7 \ 9)$$

Sea:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad \& \quad B = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \quad \& \quad w = (7 \ 9)$$

Tenemos dos matrices B y $B \in \mathbb{R}^{2 \times 2}$, mientras que el vector $w \in \mathbb{R}^2$. El producto de $A \cdot B$ nos devuelve lo siguiente:

$$AB = \begin{pmatrix} 1 \cdot 0 + 2 \cdot 2 & 1 \cdot 1 + 2 \cdot 3 \\ 3 \cdot 0 + 4 \cdot 2 & 3 \cdot 1 + 4 \cdot 3 \end{pmatrix} = \begin{pmatrix} 4 & 7 \\ 8 & 15 \end{pmatrix}$$

Ahora, sabemos que AB tiene dimensión $(2, 2)$, mientras que w tiene dimensión $(2, 1)$. Aplicando broadcasting para sumar, w se va a replicar virtualmente por filas para tener ahora la dimensión $(2, 2)$. De esa forma, pasamos de la ecuación del enunciado a la siguiente:

$$\begin{pmatrix} 4 & 7 \\ 8 & 15 \end{pmatrix} + \begin{pmatrix} 7 & 9 \\ 7 & 9 \end{pmatrix} = \begin{pmatrix} 11 & 16 \\ 15 & 24 \end{pmatrix}$$

Código en Numpy

Listing 1: La descripción del código fue generada con ayuda de GPT

```
import numpy as np

A = np.array([[1, 2],
              [3, 4]])
B = np.array([[0, 1],
              [2, 3]])
v = np.array([7, 9])

AB = A @ B
R1 = AB + v
R2 = AB + np.broadcast_to(v, AB.shape)

print("AB shape:", AB.shape, "\nAB:\n", AB)
print("v shape:", v.shape)
print("Result R1:\n", R1)
print("Result R2:\n", R2)

"""

This code shows how broadcasting works in NumPy:

1. Compute AB as a 2x2 matrix product.
2. v has shape (2,), NumPy interprets it as (1,2).
   When added to AB (2,2), it is automatically expanded by rows (R1).
3. np.broadcast_to(v, AB.shape) explicitly forces a (2,2) view.
4. R1 and R2 are numerically identical, but internally R2
   uses a broadcasted read-only view sharing memory with v.

Summary: broadcasting allows arrays with different shapes
to be virtually expanded for efficient element-wise operations.
"""
```

Ejercicio #2 | Redes Neuronales Multicapa

Considera las redes multicapa (con funciones de activación lineal), que se muestran en la figura 1:

- Describe una ventaja (al menos) de la red A sobre la red B.
- Describe una ventaja (al menos) de la red B sobre la red A.

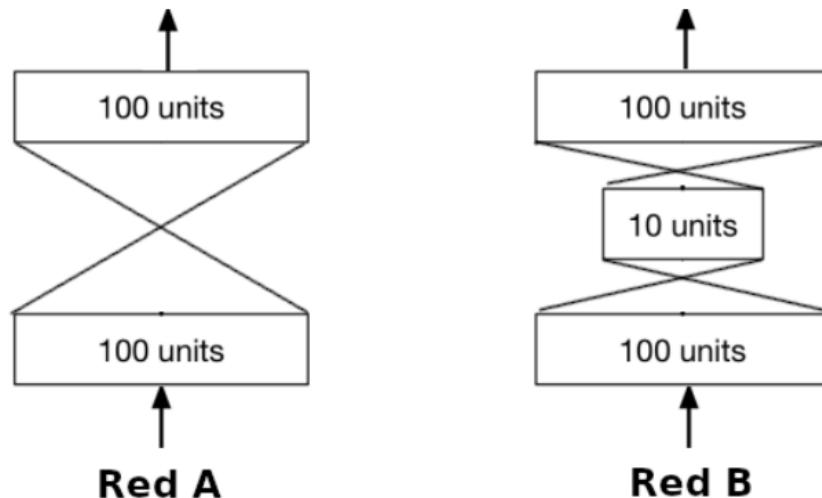


Figura 1: Esquema de dos redes neuronales.

Ventajas de la Red A

Nuestra Red A es esencialmente una transformación lineal completa entre dos espacios de dimensión 100. Hay que tomar en cuenta que la activación de nuestras capas son lineales, por lo que la red puede representar cualquier transformación lineal entre entrada y salida en el mismo espacio. Lo anterior provoca que no se pierda información o detalles de las características de nuestros vectores de entrada.

Sin embargo, la Red A tiene una desventaja bastante crítica para la mayoría de las tareas. Al pasar la información de manera directa, de los 100 nodos a los otros 100 nodos, se corre el riesgo de sobreajustar el entrenamiento a nuestros datos y no llegar a generalizar en absoluto.

Ventajas de la Red B

Ahora bien, la Red B introduce un cuello de botella de 10 neuronas entre capas de dimensión 100. Esto provoca un efecto de regularización, pues al reducir la dimensionalidad en dicha capa intermedia, se fuerza al modelo a aprender representaciones latentes comprimidas que capturen las características más importantes de nuestros datos. Podríamos pensar en que tal efecto es similar al de una proyección de Análisis de Componentes Principales (PCA), en donde el cuello de botella concentra la mayor varianza explicativa de nuestros datos.

Justo esa arquitectura con una capa en medio es lo que vuelve a la Red B a una más robusta en términos de generalización. A diferencia de la Red A, esta corre menos riesgo de sobreajustar ya que no pasa de manera directa los datos de la capa de 100 nodos a la otra de 100 nodos. El modelo se queda con un “mapeo de características” reteniendo los patrones estructurales de nuestros datos. Claro que con sus limitaciones, pues después de todo se trata de una red sencilla.

Matemáticamente, esto se vería algo así:

$$\text{La Red B se implementa como: } f(x) = W_2 W_1 x$$

En este caso:

- $W_1 \in \mathbb{R}^{10 \times 100}$ y proyecta x a un subespacio de dimensión 10.
- $W_2 \in \mathbb{R}^{100 \times 10}$ reconstruye desde el anterior subespacio a dimensión 100.

Por lo tanto, la matriz de la red se representaría de la siguiente manera:

$$M_{\text{Red B}} = W_2 W_1 \in \mathbb{R}^{100 \times 100}$$

Y esto implica que:

$$\text{rank}(M) \leq \min\{\text{rank}(W_1), \text{rank}(W_2)\} \leq 10$$

La primera transformación W_1 crea un espacio intermedio cuya dimensión es como máximo de 10. La segunda transformación W_2 actúa sobre el espacio anterior, ya de por sí limitado. Por lo que la dimensión del espacio final no puede ser mayor que la del cuello de botella. Esto lleva a que $\text{rank}(W_2 W_1) \leq \text{rank}(W_1)$. Creo que con lo anterior queda demostrado que la Red B no puede representar cualquier transformación lineal arbitraria en \mathbb{R}^{100} , solo aquellas de rango bajo, i.e. 10.

Podemos pensar en la tarea de filtrado de ruido en audio o imágenes, o detección de bordes. El cuello de botella va a favorecer la extracción de propiedades generales en lugar de detalles específicos que sometan a nuestro modelo a sobreajustar.

Ejercicio #3 | Regularización L^1

Considera la regularización L^1 de una función de costo L , que asumimos es continuamente diferenciable:

$$\tilde{L}(w; X, y) = L(w; X, y) + \alpha \sum_i |w_i|$$

- (a) Como en clase, considera una aproximación de segundo orden alrededor de \mathbf{w}^* y muestra que la aproximación regularizada de \tilde{L} es:

$$\hat{L}(w) = L(w^*) + \sum_i \left(\frac{1}{2} H_{ii} (w_i - w_i^*)^2 + \alpha |w_i| \right)$$

donde se asume que los datos están decorrelacionados (i.e. “blanqueados”), tal que \mathbf{H} es un matriz diagonal con $H_{ii} > 0$.

Aproximación Cuadrática para Regularización L^1

Sea $L : \mathbb{R}^d \rightarrow \mathbb{R}$ una función dos veces diferenciable y w^* un minimizador local de L . La expansión de Taylor de segundo orden alrededor de w^* es:

$$L(w) = L(w^*) + \nabla L(w^*)^\top (w - w^*) + \frac{1}{2} (w - w^*)^\top H(w^*) (w - w^*) + O(||w - w^*||^2)$$

Como w^* es el óptimo, entonces $\nabla L(w^*) = 0$, por lo que podemos despreciar el término que lo acompaña y nos quedamos con lo siguiente:

$$L(w) \approx L(w^*) + \frac{1}{2} (w - w^*)^\top H(w^*) (w - w^*)$$

Después de lo anterior, la pérdida regularizada con L^1 se va a definir como:

$$\hat{L}(w) = L(w) + \alpha ||w||_1 = L(w) + \alpha \sum_{i=1}^d |w_i|$$

Por lo tanto, nuestra expresión queda de la siguiente manera:

$$L(w) = L(w^*) + \frac{1}{2}(w - w^*)^\top H(w^*)(w - w^*) + \alpha \sum_{i=1}^d |w_i|$$

Ahora, como se no sindica que los datos son decorrelacionados, o blanqueados, el Hessiano evaluado en w^* es diagonal o aproximadamente diagonal. En ese sentido, podemos argumentar que nuestra curvatura queda descrita como $H(w^*) = \text{diag}(H_{11}, H_{22}, \dots, H_{dd})$ con $H_{ii} > 0$. Bajo esta suposición, el término cuadrático se simplifica:

$$(w - w^*)^\top H(w^*)(w - w^*) = \sum_{i=1}^d H_{ii}(w_i - w_i^*)^2$$

Esto nos permite llegar justo a lo que buscábamos, haciendo el ajuste:

$$L(w) = L(w^*) + \sum_{i=0}^d \left(\frac{1}{2} H_{ii}(w_i - w_i^*)^2 + \alpha |w_i| \right)$$

Una de las conclusiones a las que podemos llegar al estudiar regularización L^1 es que el término cuadrático $\frac{1}{2} H_{ii}(w - w^*)^2$ penaliza la desviación o alejamiento del punto w^* , con una rigidez dada por la curvatura H_{ii} y el término $\alpha |w_i|$ fuerza que los coeficientes se acerquen a cero. De igual forma, $(w_i - w_i^*)^2$ es la distancia que hay entre cierto peso y el óptimo w^* , al ser un término cuadrado y definido positivo, cualquier incremento es sumamente penalizado.

(b) Muestra que \hat{L} se minimiza en:

$$w_i = \text{sign}(w_i^*) \max \left\{ |w_i^*| - \frac{\alpha}{H_{ii}}, 0 \right\}$$

¿En qué casos tendremos soluciones *sparse*, i.e. cuando $w_i = 0$?

Sparcity provocado por L^1

Ya que tenemos la aproximación dada en el inciso anterior:

$$L(w) = L(w^*) + \sum_{i=0}^d \left(\frac{1}{2} H_{ii}(w_i - w_i^*)^2 + \alpha |w_i| \right)$$

Y tomando en cuenta que consideramos a H diagonal, entonces el problema se puede descomponer. Lo que queremos es minimizar para cada i-ésimo peso:

$$\phi_i(w_i) = \frac{1}{2}H_{ii}(w_i - w_i^*)^2 + \alpha|w_i| \quad \text{con } H_{ii} > 0$$

Esto nos está indicando que la función es estrictamente convexa. Así, recurrimos a condiciones de subgradiente para poder trabajar con $|w_i|$ ya que no es continuamente diferenciable. Tenemos entonces:

$$\begin{cases} \{+1\} & \text{si } w_i > 0 \\ [-1, +1] & \text{si } w_i = 0 \\ \{-1\} & \text{si } w_i < 0 \end{cases}$$

La condición de optimalidad para el mínimo es: $0 \in \partial\phi_i(w_i)$, lo que provoca:

$$0 \in H_{ii}(w_i - w_i^*) + \alpha\partial|w_i|$$

Ahora, vamos estudiando por casos.

Caso 1: $w_i > 0$

Para este caso, sucede lo siguiente:

$$0 = H_{ii}(w_i - w_i^*) + \alpha \Rightarrow w_i = w_i^* - \frac{\alpha}{H_{ii}}$$

Caso 2: $w_i < 0$

$$0 = H_{ii}(w_i - w_i^*) - \alpha \Rightarrow w_i = w_i^* + \frac{\alpha}{H_{ii}}$$

Caso 3: $w_i = 0$

Para este caso, la condición de optimalidad es $0 \in H_{ii}(0 - w_i^*) + \alpha[-1, +1]$, lo que se traduce en lo siguiente:

$$H_{ii}w_i^* \in \alpha[-1, +1] \Rightarrow |w_i| \leq \frac{\alpha}{H_{ii}}$$

Si se cumple la condición, entonces $w_i = 0$ y eso es el mínimo.

Ahora bien, juntando los casos anteriores, nos queda lo siguiente:

$$w_i = \text{sign}(w_i^*) \max \left\{ |w_i^*| - \frac{\alpha}{H_{ii}}, 0 \right\}$$

A esta regla se le conoce como “Regla de Soft-Thresholding”. Hablando de *sparsity*, habrá ceros cuando en la coordenada i se cumpla que:

$$|w_i^*| \leq \frac{\alpha}{H_{ii}}$$

Es decir, cuando el umbral $\frac{\alpha}{H_{ii}}$ sea superado o igualado en magnitud del coeficiente no regularizado w_i^* . Ahora bien, con un $\alpha \uparrow$ se crea un umbral mayor y más coeficientes se anulan. Con una curvatura pequeña, i.e. un $H_{ii} \downarrow$, se amplia el umbral y hay más coeficientes anulados.

Ejercicio #4 | Backpropagation

Considera un problema de clasificación multiclase y una red neuronal densamente conectada con una capa oculta, como se muestra en la figura 2. Considera también la función Sigmoide como activación de las unidades ocultas, la función Softmax para la estimación de las capas finales y a Cross-Entropy como función de costo.

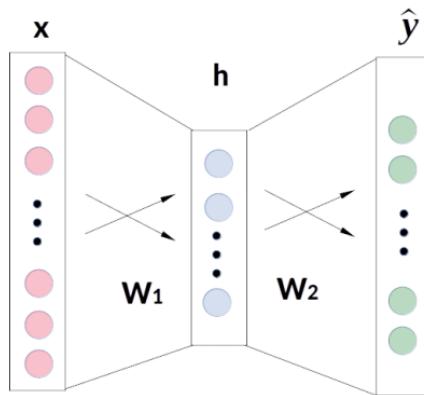


Figura 2: Red neuronal densamente conectada con una sola capa oculta.

Teoría

- (a) Muestra que Softmax es invarianta a traslaciones constantes del vector de entrada i.e. para cualquier vector x y cualquier constante C :

$$\text{Softmax}(x) = \text{Softmax}(x + c)$$

Donde la operación $x + c$ se realiza con broadcasting. Recuerda que:

$$\text{Softmax}(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Lo anterior es útil cuando se escoge $c = -\max(x)$ i.e. quitando el valor mayor en todos los elementos de x para mejorar la estabilidad numérica.

Para resolver este inciso, conviene primero recordar lo que es broadcasting. Este mecanismo nos permite operar arreglos de forma que se puede replicar, si hacer copias en memoria, un arreglo de menor dimensión a uno de mayor dimensión. Para el caso de $x \in \mathbb{R}^d$ ya un escalar $c \in \mathbb{R}$, la operación $x + c$ consistiría en replicar ese c a lo largo de toda la coordenada de x , tal que:

$$x + c = (x_1 + c, x_2 + c, \dots, x_d + c)$$

Recordemos también la definición de la función Softmax:

$$\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}} \quad \text{con } i = 1, 2, \dots, d$$

De ese modo, si consideramos al vector $x + c$ trasladado, entonces nos encontraremos con:

$$\text{Softmax}(x + c)_i = \frac{e^{x_i+c}}{\sum_{j=1}^k e^{x_j+c}} = \frac{e^c \cdot e^{x_i}}{e^c \cdot \sum_{j=1}^k e^{x_j}} = \frac{e^{x_i}}{\sum_{j=1}^k e^{x_j}} = \text{Softmax}(x)_i$$

Esta demostración nos está diciendo que, bajo las condiciones que establecimos, Softmax solo depende de las diferencias entre los componentes en x , no de sus valores absolutos. En otras palabras, podemos decir que el mecanismo de broadcasting permite escribir al vector $x + c$ como $x + c = (x_1, x_2, \dots, x_d) + c$.

Además, se evitan valores muy grandes que se cargarían por la operación suma entre cada entrada del vector x con la constante c , algo que podría causar problemas de desbordamiento en el cómputo de la memoria.

(b) Para un escalar x , muestra que el gradiente de la función Sísmoide es:

$$\sigma(x) = (1 - \sigma(x))$$

Nosotros ya sabemos que la Sísmoide se define como:

$$\sigma(x) = \frac{1}{1 - e^{-x}}$$

Entonces, si derivamos utilizando regla de la cadena, tenemos:

$$\begin{aligned}
 \sigma'(x) &= \frac{d}{dx} [(1 - e^{-x})^{-1}] = (-1)(1 - e^{-x})^{-1-1} \cdot (-e^{-x}) \\
 &= -(1 - e^{-x})^{-2} \cdot (-e^{-x}) \\
 &= \frac{e^{-x}}{(1 - e^{-x})^2}
 \end{aligned}$$

Ahora, bien:

$$1 - \sigma(x) = 1 - \frac{1}{1 - e^{-x}} = \frac{1 - (1 - e^{-x})}{1 - e^{-x}} = \frac{e^{-x}}{1 - e^{-x}}$$

Por lo tanto:

$$\boxed{\sigma(x) \cdot (1 - \sigma(x)) = \frac{1}{1 - e^{-x}} \cdot \frac{e^{-x}}{1 - e^{-x}} = \frac{e^{-x}}{(1 - e^{-x})^2} = \sigma'(x)}$$

Lo que todo este desarrollo nos está diciendo es que podemos obtener la derivada de la sigmoide sin necesidad de clacular directamente con la regla de la cadena. Es decir, podemos definirla solo como el producto de la misma Sigmoide con ella misma. La primer conclusión que podríamos pensar respecto a este resultado es que ahorramos poder computacional durante el Backpropagation, así como ahorro en la memoria, ya que se utilizan resultados previos, haciendo que el algoritmo de retropropagación sea más eficiente. Es una construcción de la derivada de la Sigmoide que facilita la propagación del error hacia las capas ocultas.

(c) Muestra que el gradiente de la capa de salida es:

$$\frac{\partial L(y, \hat{y})}{\partial z} = \hat{y} - y$$

Donde $\hat{y} = \text{Softmax}(z)$ para algún vector z que proviene de la capa de salida. Qué interpretación puedes darle a esa expresión? LA función de costo es Cross-Entropy:

$$L(y, \hat{y}) = - \sum_i y_i \log(\hat{y})$$

Aquí y es un vector *one-hot* de las clases y \hat{y} de las probabilidades estimadas.

Sean los logits $z \in \mathbb{R}^d$, ya sabemos que Softmax se define como:

$$\hat{y}_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

Y que la pérdida es una Cross-Entropy:

$$L(y, \hat{y}) = - \sum_i y_i \log(\hat{y})$$

Ahora, como estamos trabajando con una y en *one-hot encoding*, tenemos que:

$$\frac{\partial L}{\partial z_j} = \frac{\partial L}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_j}$$

Tenemos que:

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{y}{y_i} y \quad \Rightarrow \quad \frac{\partial \hat{y}_i}{\partial z_i} = \hat{y}(\delta_{ij} - \hat{y}_j)$$

En este caso, estamos usando la delta de Kronecker, definida como:

$$\delta_{ij} = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

La probabilidad de la clase i se define como:

$$\hat{y}_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Aplicamos la regla del cociente para derivar con respecto a z_i :

$$\begin{aligned} \frac{\partial \hat{y}_i}{\partial z_i} &= \frac{\left(\frac{\partial}{\partial z_i} e^{z_i} \right) \left(\sum_k e^{z_k} \right) - e^{z_i} \left(\frac{\partial}{\partial z_i} \sum_k e^{z_k} \right)}{\left(\sum_k e^{z_k} \right)^2} \\ &= \frac{e^{z_i} \left(\sum_k e^{z_k} \right) - e^{z_i} (e^{z_i})}{\left(\sum_k e^{z_k} \right)^2} \end{aligned}$$

Separando la fracción:

$$\begin{aligned} &= \frac{e^{z_i}}{\sum_k e^{z_k}} - \frac{e^{z_i} e^{z_i}}{\left(\sum_k e^{z_k} \right)^2} \\ &= \frac{e^{z_i}}{\sum_k e^{z_k}} \left(1 - \frac{e^{z_i}}{\sum_k e^{z_k}} \right) \end{aligned}$$

Lo que simplifica a:

$$= \hat{y}_i (1 - \hat{y}_i)$$

Ahora, derivamos \hat{y}_i con respecto a una entrada diferente, z_j . La variable z_j solo aparece en el denominador.

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{\left(\frac{\partial}{\partial z_j} e^{z_i} \right) \left(\sum_k e^{z_k} \right) - e^{z_i} \left(\frac{\partial}{\partial z_j} \sum_k e^{z_k} \right)}{\left(\sum_k e^{z_k} \right)^2}$$

Como $i \neq j$, la derivada del numerador es cero:

$$\begin{aligned} &= \frac{0 \cdot \left(\sum_k e^{z_k} \right) - e^{z_i} (e^{z_j})}{\left(\sum_k e^{z_k} \right)^2} \\ &= -\frac{e^{z_i} e^{z_j}}{\left(\sum_k e^{z_k} \right)^2} \end{aligned}$$

Reagrupando los términos:

$$= -\left(\frac{e^{z_i}}{\sum_k e^{z_k}} \right) \left(\frac{e^{z_j}}{\sum_k e^{z_k}} \right)$$

Lo que simplifica a:

$$= -\hat{y}_i \hat{y}_j$$

Podemos escribir ambos casos en una sola expresión usando el delta de Kronecker, δ_{ij} :

$$\frac{\partial \hat{y}_i}{\partial z_j} = \hat{y}_i (\delta_{ij} - \hat{y}_j)$$

Donde δ_{ij} es 1 si $i = j$ y 0 en caso contrario.

Si $i = j$: $\delta_{ij} = 1$, entonces la expresión se convierte en:

$$\frac{\partial \hat{y}_i}{\partial z_j} = \hat{y}_i (1 - \hat{y}_j)$$

Si $i \neq j$: $\delta_{ij} = 0$, entonces la expresión se convierte en:

$$\frac{\partial \hat{y}_i}{\partial z_j} = \hat{y}_i (0 - \hat{y}_j) = -\hat{y}_i \hat{y}_j$$

Entonces:

$$\frac{\partial L}{\partial z_j} = \sum_i \left(-\frac{y_i}{\hat{y}_i} \right) \hat{y}_i (\delta_{ij} - \hat{y}) = \sum_i \left(-y_i(\delta_{ij} - \hat{y}_i) \right) = -y_j + \hat{y}_j \sum_i y_i$$

Como tenemos y en *one-hot*, tenemos que $\sum_i y_i = 1$:

$$\frac{\partial L}{\partial z_j} = \hat{y}_j - y_j \quad \Rightarrow \quad \nabla_z L = \hat{y} - y$$

De todo lo anterior, podemos observar que Softmax convierte logits en probabilidades, Cross-Entropy penaliza desviaciones respecto a y y que el gradiente $\hat{y} - y$ indica cuánto tenemos de probabilidad por clase.

(d) Considerando los incisos anteriores, obtén los gradientes respecto a los parámetros del modelo calculando:

$$\left\{ \frac{\partial(y, \hat{y})}{\partial x} \right\}$$

Para, de esa manera, obtener las ecuaciones de Backpropagation de la red. Recuerda que el paso Forward calcula las activaciones:

$$h = \sigma(w_1x + b_1)$$

$$\hat{y} = \text{Softmax}(w_2h + b_2)$$

Recordemos también que la función de activación en un vector se aplica entrada por entrada.

Creo que es buena idea comenzar hablando un poco sobre el proceso de retropropagación y cómo es tan fundamental para el desarrollo de las Redes Neuronales. Backpropagation consiste en la optimización iterativa de los parámetros internos de la red neuronal, i.e. los pesos y los sesgos. Lo que se busca es minimizar la función de pérdida que se encarga de cuantificar el error de las predicciones y los valores verdaderos. Este mecanismo se trata entonces de un algoritmo fundamental que permite el proceso de aprendizaje, claculando de manera eficiente el gradiente de la función de pérdida con respecto a cada parámetro de la red. Intentaremos desglozar el proceso.

Ya sabemos que el Forward Pass es el proceso en el cual la red transforma una entrada x en una predicción y . Cada entrada x se procesa en una transformación lineal utilizando la matriz de pesos W_1 y el vector de sesgos b_1 de la capa de entrada. A este resultado se le suele llamar “vector de preactivaciones”, a_n .

$$a_1 = W_1 x + b_1$$

Al vector de preactivaciones se le pasa hacia una función de activación no lineal, en nuestro caso una Sigmoide.

$$h = \sigma(a_1)$$

Las activaciones h de la capa oculta son procesadas por una segunda transformación lineal, utilizando ahora los parámetros W_2 y b_2 , para generar así un vector de puntuaciones z :

$$z = W_2 h + b_2$$

Finalmente, el vector z se le normaliza mediante una Softmax, que lo transforma en una distribución de probabilidad multinomial, \hat{y} , donde cada elemento representea la probabilidad predicha por cada clase:

$$\hat{y} = \text{Softmax}(z)$$

Ahora bien, la cuantificación de la pérdida. Sabemos que la discrepancia entre la predicción \hat{y} y la etiqueta y , en *one-hot*, se cuantifica mediante la función de pérdida Cross-Entropy categórica L . Esta función mide la divergencia de información entre las dos distribuciones:

$$L = - \sum_i y_i \log(\hat{y})$$

El objetivo del aprendizaje es, entonces, minimizar el valor de L a través del ajuste de los parámetros:

$$\theta = \{W_1, b_1, \dots, W_n, b_n\}$$

El cálculo de los gradientes de la función de pérdida L con respecto a los parámetros W_2 y b_2 de la capa de salida mediante la aplicación de la regla de cadena. Partimos de la expresión conocida de la derivada de la pérdida con respecto al vector de puntuaciones z :

$$\delta = \frac{\partial L}{\partial z} = \hat{y} - y$$

Aquí, \hat{y} es el vector de probabilidades predichas y y es el vector de etiquetas reales. Así, definimos z como: $z = W_2 h + b_2$ con h como el vector de observaciones de la capa oculta con dimensión $H \times 1$ y W_2 con dimensiones $C \times H$. En este caso, tenemos C como

la dimensión de las clases y H como la dimensión o cantidad de neuronas por capa oculta. Entonces:

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial W_2}$$

Dado que $\frac{\partial L}{\partial z} = \delta$, hay que concentrarse en $\frac{\partial z}{\partial W_2}$. Calculamos la derivada de L con respecto a cada elemento de W_{2ij} de W_2 :

$$z_i = \sum_H^{k=1} W_{2ik} h_k + b_{2i}$$

$$\frac{\partial z_i}{\partial W_{2ij}} = h_j \quad \text{para } k \neq j \quad \frac{\partial z_i}{\partial W_{2ij}} = 0$$

$$\frac{\partial L}{\partial W_{2ij}} = \sum_{k=1}^C \frac{\partial L}{\partial z_k} \cdot \frac{\partial z_k}{\partial W_{2ij}} = \delta h^\top$$

Por lo tanto:

$$\frac{\partial L}{\partial W_2} = \delta h^\top$$

Calculando de una manera similar a $\frac{\partial L}{\partial b_2}$:

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial b_2}$$

Entonces, para cada elemento b_{2i} de b_2 , tenemos:

$$\frac{\partial z_i}{\partial b_{2i}} = 1$$

Y para cada $k \neq i$ tenemos:

$$\frac{\partial z_i}{\partial b_{2i}} = 0 \quad \text{entonces} \quad \frac{\partial z_i}{\partial b_{2i}} = \sum_{k=1}^C \frac{\partial L}{\partial z_k} \frac{\partial z_k}{\partial b_{2i}} = \frac{\partial L}{\partial z_i} \cdot 1 = \delta$$

Por lo tanto:

$$\frac{\partial L}{\partial b_2} = \delta = \hat{y} - y$$

Los gradientes de la pérdida L con respecto a los parámetros de la capa de salida son:

$$\boxed{\frac{\partial L}{\partial W_2} = \delta h^\top \quad \frac{\partial L}{\partial b_2} = \delta}$$

Estos resultados muestran cómo el error en *delta* se propaga hacia atrás, a través de la red, utilizando las activaciones h de la capa oculta para ajustar los pesos W_2 y directamente asignando el error a los sesgos b_2 .

Una vez calculados los gradientes de la salida, se procede a propagar el error hacia la capa oculta. Este proceso implica calcular el gradiente de la pérdida L con respecto a las activaciones a_1 de la capa oculta, denotando como $\delta^h = \frac{\partial L}{\partial a_1}$, y luego utilizar este valor para obtener los gradientes de los parámetros W_1 y b_1 .

Las activaciones h se obtienen aplicando la Sigmoide de las preactivaciones a_1 , i.e. $h = \sigma(a_1)$. Para encontrar δ^h , usamos regla de la cadena:

$$\delta^h = \frac{L}{\partial a_1} = \frac{\partial L}{\partial h} \cdot \frac{\partial h}{\partial a_1}$$

La derivación de la Sigmoide es $\sigma'(a_1) = \sigma(a_1)(1 - \sigma(a_1)) = h \odot (1 - h)$. Aquí \odot se define como el producto de Hadamard, i.e. elemento a elemento.

Dado que h es un vector, $\frac{\partial h}{\partial a_1}$ es una matriz diagonal con elementos $h_i(1 - h_i)$. Al multiplicar por el vector $\frac{\partial L}{\partial h}$, esto equivale a un producto elemento a elemento:

$$\delta^h = \frac{\partial L}{\partial h} \odot \sigma'(a_1) = (W_2^\top \delta) \odot (h \odot (1 - h))$$

En este caso, δ^h es de dimensión $H \times 1$, y se trata del error propagado hacia las preactivaciones de la capa.

Ahora bien, el cálculo de los gradientes de los parámetros de la capa de entrada con W_1 y b_1 .

Tenemos $\delta^h = \frac{\partial L}{\partial a_1}$. Calculamos los gradientes para W_1 y b_1 . Las preactivaciones se definen como: $a_1 = W_1 x + b_1$. Entonces:

$$\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a_1} \cdot \frac{\partial a_1}{\partial W_1} = \delta^h \frac{\partial a_1}{\partial W_1} = \delta^h x^\top$$

Así:

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial a_1} \cdot \frac{\partial a_1}{\partial b_1} = \delta^h$$

Así, Backpropagation se encarga de mover el error hacia atrás de la salida. Este mecanismo es esencial para el ajuste iterativo de los pesos y sesgos del modelo, intentando mejorar tras cada época los valores para obtener una mejor predicción en la salida.

Resumen de código

Resultados

Conclusiones

Ejercicio #1 | Descripción del corpus

Analiza el corpus y reporta:

- Número de documentos, tokens y vocabulario.
- Hapax legomena y su proporción.
- Porcentaje y su proporción.
- Estadísticas por clase (número de documentos, tokens y vocabulario).

Teoría

Resumen de código

Resultados

Conclusiones

“Una nota personal sobre música, sonido y electrónica”

Haz un resumen corto del ensayo: “*Una nota personal sobre música, sonido y electrónica*”

El ensayo de Daphne Oram comienza con una pregunta: “¿Qué es la memoria?” Es justo esa cuestión la que abre el camino para que Oram presente su perspectiva sobre cómo es que funciona nuestra mente y, valga la redundancia, nuestra memoria. Y es que para Oram no es improbable que nuestra memoria funcione como una especie de máquina reproductora de sonido (o de señales), al estilo de una grabadora de cintas.

Las analogías que hace Oram tanto para explicar su idea de las memorias como el comportamiento humano se centran en la música. Para ella, hay un proceso de retroalimentación que impulsa la resonancia de ciertas memorias y secciones del cerebro. Uno de los primeros ejemplos que pone es el del estado de ánimo por las mañanas, argumentando que cuando estamos de buen humor los problemas parecen más livianos, mientras que cuando estamos de malas, todo parece peor de lo que en realidad es. Menciona que: “los sonidos musicales que se reproducen a través del proceso de retroalimentación por cinta cambian de la misma manera; es decir, cambian de acuerdo a los ajustes de control de tono (o filtros) en el circuito de retroalimentación”.

Interpreto lo anterior como que aquellas máquinas que procesan el sonido de los instrumentos, como un pedal para una guitarra eléctrica, o incluso la afinación de un piano o cualquier otro instrumento, determinan la calidad del sonido y cómo el escucha los puede apreciar. Un violín mal afinado logrará desconcertar al público, así como un pedal llevado al límite puede producir sonidos no muy agradables para todos. La analogía es clara: los estados mentales son como la afinación o procesamiento de nuestros pensamientos, lo cual afecta directamente cómo nos sentimos a lo largo del día y cómo interpretamos los problemas.

Oram también menciona que la grabación de recuerdos más pura y duradera es la de los recuerdos de la infancia. Según ella, “fueron grabados en una cinta virgen”. Tanto recuerdos buenos como malos en aquella etapa de la vida seguramente perdurarán a lo largo del tiempo porque les pusimos especial atención o porque fueron tan intensos que quedaron “sobrecargados cuando se grabaron”.

Bajo su óptica, no es como que la memoria sea una biblioteca en la cual existan cintas con los recuerdos grabados (esto refleja un poco de la época en la que se escribió el ensayo). Para Oram, la memoria es como un circuito de retroalimentación en el cual se emiten

ecos continuos de señales hasta que se acaba la energía. Esto provoca que, cuando estemos recordando algo, haya resonancia de tal manera que se vayan alimentando ciertas partes del circuito relacionadas con ciertos temas (como si se tratara de tópicos). Interpreto que esto es algo así: si comienzo a pensar en alguien de mi pasado, quizás recuerde con nostalgia, felicidad, enojo o tristeza a esa figura. Dependiendo de mi afinación inicial (estado de ánimo) y del recuerdo en particular que esté proyectando, se generará una señal que probablemente alimentará o destruirá otras señales. Se pueden comenzar a activar otras zonas del circuito que se relacionan con esa persona, como los lugares a los que fuimos, las emociones que se activaron en el recuerdo, incluso cosas como recuerdos de temperatura, olores, o quizás hasta sonidos como tal.

« Al clasificar las señales de la memoria, los mismos circuitos no se moverían de un área a otra en el cerebro, sino que la energía, por mera inducción simpática, se encontraría resonando con células de retroalimentación del mismo tipo en paralelo con aquellas ya ocupadas con el mismo tipo de información. Entonces, un pensamiento de memoria podría resonar en muchas células de retroalimentación en múltiples partes del cerebro porque tiene afinidad con los pensamientos almacenados en esas regiones. »

Dependiendo de la intensidad de la señal, que Oram llama algo así como “retroalimentación de la memoria”, se provocarán resonancias que hagan crecer otros tópicos y comiencen a crecer en magnitud. Esto nos lleva a algo que Oram aborda casi al final de su ensayo: los estados mentales que nos hunden en ansiedad, depresión o cosas similares (aunque Oram no menciona dichas definiciones, creo que es claro que se refiere a ello).

Además, también argumenta que la memoria no es perfecta. Oram argumenta que el cerebro probablemente prefiere no esforzarse por recordar todo a detalle, sino que hay una parte del mismo que se encarga de “generar” ciertos detalles, a la cual le llama: “departamento de racionalización”. Creo que esto se relaciona un poco con la IA generativa, pues no es que los algoritmos generativos se encarguen de entender y procesar lo que van a crear, sino que calculan lo más probable. Es decir, proyectan para los videos, los textos, las imágenes o los sonidos lo que es más probable que siga después de su último estado. Así, el cerebro puede quedarse con una proyección en baja dimensión de los recuerdos, y de esa manera llenar los huecos con lo que es más probable que sucedió bajo ese contexto.

Quizás el último cuarto del ensayo es el que más me interesó. En esta sección, Oram comienza hablando acerca de los “estados de retroalimentación sin control”. Menciona que el cerebro humano, bajo la óptica de que se puede describir como una combinación de señales de sonido, puede colapsar si no se controla apropiadamente el volumen de reproducción. “Esta sobrecarga que se da por subir demasiado el volumen de la reproducción ciertamente es algo en lo que vale la pena reflexionar: parece ser la perdición de nuestra existencia”. Oram continúa diciendo que si insistimos en subir demasiado el volumen, los problemas (pensamientos) que estemos amplificando solo se volverán más grandes en nuestra mente:

“cada simple onda sinusoidal termina convirtiéndose en ruido blanco”. Al permitir la resonancia de los pensamientos negativos sin control alguno, terminaremos convirtiéndolos en una especie de “muro de sonido”, pero no en el buen sentido, sino en uno que nos abruma, perturba y nos deja en un estado mental peor al cual en el que iniciamos, “entre más te concentres en esos pensamientos, más empeoran”.

Para Oram es claro que en el caso de un reproductor musical es tan sencillo como bajar el volumen. Sin embargo, también tiene claro que el cerebro y la mente humana son mucho más complejos que eso. No basta con intentar “bajar el volumen” a la ansiedad, a los pensamientos intrusivos. Una primera aproximación es llevar el discurso interno a otro tema, “con la esperanza de que ese nuevo tema no sobrecargue el sistema de grabación con los mismos terribles resultados que antes”.

Una segunda opción es la terapia de choque. Si dejamos que el cerebro se sobrecargue, seguramente en algún momento la misma intensidad de los pensamientos nos lleve a la eliminación repentina y violenta de esa sensación negativa que nos provoca. Me pregunto si Oram se estaba refiriendo quizás a la despersonalización o disociación de la mente con respecto a esos problemas. Quiero entender que para Oram esta no era la mejor opción, pues llevar al límite la mente puede resultar muy mal, no solo a nivel personal, sino también social. Llevarse al límite puede terminar en las peores decisiones.

La tercera opción, y la que para Oram es todo un alivio, es un “sencillo remedio al que me refiero: cuando un pensamiento te preocupa demasiado, ve y díselo a alguien, o escríbelo en una carta. Tal vez la razón por la que esto es un remedio es que el pensamiento pierde su energía cuando se transduce a la palabra hablada o escrita”. Sacar lo que lleva cargando tu mente siempre funciona como un regulador de presión, una válvula de escape que desinflama un poco el dolor o pesadez con la cual vamos andando por el mundo, mental y físico.

Para Oram esta tercera opción es maravillosa porque los pensamientos se someten a una especie de censura que filtra las sensaciones. Además de que cambia la perspectiva y “el patrón de onda personal” que eras cuando lo pensabas es distinto al que eres cuando lo compartes, lo hablas o lo escribes. Puede interpretarse como ver con mayor perspectiva las cosas, o como una intuición musical: “si escuchamos veinte violines interpretando la misma frase musical al unísono [...] el sonido resultante tiene una cualidad placentera que no está presente cuando la misma frase musical, diseñada para veinte intérpretes, se toca por un mero violín”. Oram menciona que un solo para violín puede ser muy rígido, pero con varios acompañándolo, su sonido se vuelve más acogedor.

Creo que puedo interpretar esto también como una especie de cambio de dimensionaldad. Imagino los pensamientos como existiendo en alta dimensionalidad: nos afectan a nivel físico, anímico y mental. Si dejamos habitando solo en nuestra conciencia a los pensamientos intrusivos y negativos, y no los procesamos de alguna manera, terminarán envol-

viéndonos de manera casi invisible. Sin embargo, al traducirlos a algo como la escritura, los bajamos a una dimensión más pequeña, los encapsulamos en algo casi unidimensional como lo son las palabras plasmadas en una pantalla o en una hoja de papel. Cuando son palabras escritas, no nos pueden hacer daño a menos que los dejemos volver a la mente de inmediato. Si los hablamos con alguna otra persona, también reducimos su dimensión, y aún mejor que con lo escrito, pues las palabras no perduran en el espacio, se difuminan casi al instante de decirlas, y el acompañamiento de la familia o los amigos siempre nos dará un confort aún más grande. Entonces, reducir la dimensión de los pensamientos y quedarse con las características más importantes nos ayuda a procesarlos para vivir un día a día un poco más tranquilo.

Como alguien ansioso y acostumbrado a una tormenta de pensamientos constante, encontré esta lectura muy liberadora. Justo me ayudó a ver con otra óptica las cosas y fue una descarga para pensar en que se puede salir de los ciclos de retroalimentación negativa. Investigué un poco sobre Daphne Oram; es muy interesante saber que fue una de las primeras en introducir la música electrónica en Reino Unido y que creó una máquina llamada Oramics para traducir figuras trazadas a sonido. Sin duda debió haber sido una persona muy interesante de conocer.